Treebank Conversion based Self-training Strategy for Parsing

Zhiguo Wang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
{zgwang, cqzong}@nlpr.ia.ac.cn

Abstract

In this paper, we propose a novel self-training strategy for parsing which is based on Treebank conversion (SSPTC). In SSPTC, we make full use of the strong points of Treebank conversion and self-training, and offset their weaknesses with each other. To provide good parse selection strategies which are needed in self-training, we score the automatically generated parse trees with parse trees in source Treebank as a reference. To maintain the constituency between source Treebank and conversion Treebank which is needed in Treebank conversion, we get the conversion trees with the help of self-training. In our experiments, SSPTC strategy is utilized to parse Tsinghua Chinese Treebank with the help of Penn Chinese Treebank. The results significantly outperform the baseline parser.

1 Introduction

Syntax parsing is one of the most fundamental tasks in natural language processing (NLP) and has attracted extensive attention during the past few decades. In statistical area, according to the type of data used in training stage, the parsing approaches can be classified into three categories: supervised, semi-supervised and unsupervised. In supervised parsing approach, a high-performance parser can be built when given sufficient labeled data (Charniak, 2000; Collins, 2003; Henderson, 2004). The semi-supervised approach utilizes some labeled data to annotate unlabeled data, then uses the annotated data to improve original model, e.g., self-training (McClosky et al., 2006) and co-training (Hwa et al., 2003). In unsupervised parsing, the labeled data was not employed and all annotations and grammars are discovered automatically from unlabeled data.

State-of-the-art supervised parsers (Charniak, 2000; Collins, 2003; Henderson, 2004) require large amounts of manually annotated training data, such as the Penn Treebank (Marcus et al., 1993), to achieve high performance. However, it is quite costly and time-consuming to create high quality labeled data. So it becomes a key bottleneck for supervised approach to acquire sufficient labeled training data. Self-training is an effective strategy to overcome this shortage. It tries to enlarge the training set with automatically annotated unlabeled data and trains a parser with the enlarged training set.

During the last few decades, many Treebanks annotated with different grammar formalisms are released (Zhou, 2004; Xue et al., 2005). Although they are annotated with different schemes, they have some linguistic consistency in some extent. Intuitively, we can convert Treebank annotated with one grammar formalisms into another Treebank annotated with grammar formalism that we are interested in. For simplicity, we call the first source Treebank, and the second target Treebank. And we call this strategy as Treebank conversion.

Although both self-training and Treebank conversion can overcome the limitation of labeled data shortage for supervised parsing in some extent, they all have drawbacks. For self-training, the quality of automatically annotated unlabeled data will affect the performance of semi-supervised parsers highly. For example, McClosky et al. (2006) shows that when the parser-best list is used for self-training, the parsing performance isn't improved, but after using reranker-best list, the retrained parser achieves an absolute 1.1% improvement. For Treebank conversion, different types among Treebanks make the converting procedure very complicated, and it is very hard to get a

Proceedings of CIPS-SIGHAN Joint Conference on Chinese Language Processing (CLP2010), pages 326–333,
Beijing, August 2010

conversion Treebank constituent with target Treebank.

To overcome the limitations mentioned above, we propose a Treebank conversion based self-training strategy for parsing, which tries to combine self-training and Treebank conversion together.

Remainder of this paper is organized as follows. In Section 2, we introduce some related work. Section 3 describes details of our SSPTC strategy. In Section 4, we propose a head finding method for Task21 in CLP2010. The experiments and analysis is given in Section 5. The last section draws conclusions and describes the future work.

2 Related Work

With the development of statistical parsing approaches, large scale corpus has become an indispensable resource. Because of the limited amount of existing labeled training data and the hardness of constructing corpus, many strategies have been proposed and experimented to overcome the contradiction.

Self-training is one of the most successful strategies. McClosky et al. (2006) shows that self-training effectively improves the accuracy of English parsing. First, they trained a two-stage reranking parser(Charniak and Johnson, 2005) using Penn Treebank (PTB)(Marcus et al., 1993) and parsed 1,750k unlabeled sentences from North American News Text corpus (NANC). Then they combined the labeled NANC sentences with PTB together as training set and retrained the first stage of the parser. The final result got a 1.1% improvement over the previous best parser for section 23 of the Penn Treebank. Huang and Harper (2009) combined self-training into a PCFG-LA based parser both for English and Chinese. Experimental result showed that self-training contributed 0.83% absolute improvement using only 210k unlabeled sentences with a single generative parser. For the Chinese parsing, self-training contributed 1.03% absolute improvement.

Treebank Conversion is another potential strategy to reuse existing source Treebanks for the study of target grammar parsing. Wang et al. (1994) proposed a Treebank conversion algorithm for corpus sharing. They employed a parser with target grammar formalism to get *N*-best parse list for each sentence in source Treebank, selected the best conversion tree from the list using their algorithm, then inserted the conversion trees into training set, and finally retrained the parser with the enlarged training set. Experimental result shows their algorithm is effective. Collins et al. (1999) performed statistical constituency parsing of Czech on a Treebank that was converted from the Prague Dependency Treebank under the guidance of conversion rules and heuristic rules, and the final performance was also improved. Xia and Palmer (2001) proposed three methods to convert dependency trees into phrase structure trees with some hand-written heuristic rules. For acquisition of better conversion rules, Xia et al. (2008) proposed a method to automatically extract conversion rules from a target Treebank. Niu et al. (2009) tried to exploit heterogeneous Treebanks for parsing. They proposed a grammar formalism conversion algorithm to convert dependency formalism Treebank into phrase structure formalism, and did phrase structure parsing with the conversion trees. Their experiments are done in Chinese parsing, and the final performance is improved indeed.

In summary, from the existing work we are confident that the strategies of self-training and Treebank conversion are effective to improve the performance of parser.

3 Our Strategy

3.1 Parsing Algorithm

Although self-training and Treebank Conversion are effective for training set enlarging, they all have drawbacks. Self-training needs some parse selection strategies to select higher quality parsers. Treebank Conversion needs us to maintain the consistency between conversed Treebank and target Treebank. On the other hand, self-training strategy provides us a good idea to get annotated trees consistent with target grammar formalism, and the parse trees in source side provide a reference for higher quality parsers selecting. So we can combine self-training and Treebank Conversion together, use self-training strategy to get converted candidates for sentences in source Treebank, and select higher quality parses according to trees in source Treebank. We call this strategy Treebank

Conversion based Self-training, and show more details in Algorithm 1.

In Algorithm 1, target Treebank T_t and source Treebank T_s are input first (line 1). Then T_t is split into two parts: training set T_{train} and development set T_{dev} (line 3). And we train an

Algorithm 1

1: **Input**: T_t and T_s
2: ▷ initialize
3: $\{T_{train}, T_{dev}\} \leftarrow Split(T_t)$
4: $Parser_0 \leftarrow Train(T_{train}, T_{dev})$
5: ▷ *Iter* iterations
6: **for** $i \leftarrow 1 \ldots Iter$ **do**
7: $T^i_{s\to t} \leftarrow \phi$
8: **for** $k \leftarrow 1 \ldots N$ **do**
9: $ParseList_k \leftarrow Nbest(Parser_{i-1}, s_k)$
10: $\hat{p}_k = \arg\max_{p_j \in ParseList_k} Score(p_{s,k}, p_j)$
11: $T^i_{s\to t} \leftarrow \hat{p}_k$
12: $Parser_i \leftarrow Train(T_{train}, T_{dev}, T^i_{s\to t})$
13: **return** $Parser_{Iter}$

initial parser with T_{train} and T_{dev} in line 4. From line 6 to line 12, we train parsers with SSPTC strategy *Iter* times iteratively. Let $T^i_{s\to t}$ be the automatically converted Treebank from source Treebank to target Treebank grammar formalism during the *i*-th iteration. From line 8 to line 11, we try to get a conversion tree with target grammar for each of the *N* sentences in source Treebank. We get *N*-best parse list $ParseList_k$ for sentence s_k with $Parser_{i-1}$ (line 9), select the parse $\hat{p}_k$ with the highest score from $ParseList_k$ (line 10), and insert it into $T^i_{s\to t}$ (line 11). This procedure runs iteratively until all the trees in source Treebank have been converted, finally, we train a new parser $Parser_i$ with T_{train}, T_{dev} and $T^i_{s\to t}$ (line 12).

3.2 Parse selection

In line 10 of Algorithm 1, we select the highest quality parse $\hat{p}_k$ from $ParseList_k$ according to function $Score(p_s, p_{s\to t})$, where p_s denotes a tree in source Treebank and $p_{s\to t}$ denotes a conversion tree with target Treebank grammar formalism for p_s. $Score(p_s, p_{s\to t})$ compares $p_{s\to t}$ with p_s and computes a score for $p_{s\to t}$ taken p_s as a reference. According to the idea proposed in Wang et al. (1994), we use the number of aligned constituents in the source and target trees to construct $Score(p_s, p_{s\to t})$. We propose two types of $Score(p_s, p_{s\to t})$ as follows.

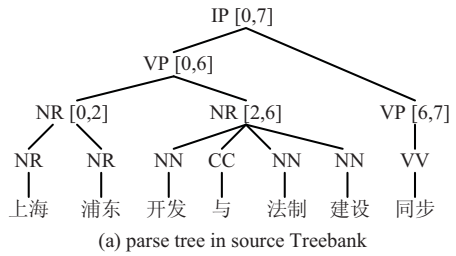
(1) Unlabeled aligned constituents F1 score (UAF)

First, we define a constituent as *tag[i,j]*, which represents a non-terminal node labeled with *tag* and spanning words from positions *i* to *j* of the input sentence. A non-terminal node in $p_{s\to t}$ aligns with a non-terminal node in p_s when they span the same words. If two nodes are aligned, we call them an aligned constituent and denote the aligned relationship as $tag_s[i,j] \Leftrightarrow tag_t[i,j]$. For example in Figure 1, there are three aligned constituents between the source Treebank tree and the conversion tree, and we can denote them as $IP_s[0,7] \Leftrightarrow dj_t[0,7]$, $NR_s[0,2] \Leftrightarrow sp_t[0,2]$ and $NR_s[2,6] \Leftrightarrow np_t[2,6]$, respectively.

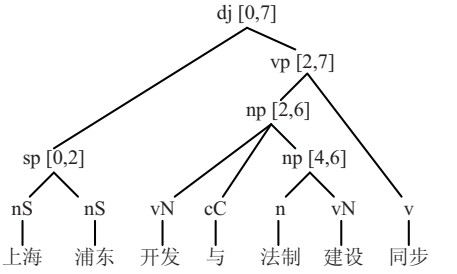
When given p_s and $p_{s\to t}$, we can easily collect all the aligned constituents. So we define Unlabeled aligned constituents Precision (UAP) and Unlabeled aligned constituents Recall (UAR) as follows.

$$UAP = \frac{\sum_{i,j} Count(tag_s[i,j] \Leftrightarrow tag_t[i,j])}{\sum_{i,j} Count(tag_t[i,j])}$$

$$UAR = \frac{\sum_{i,j} Count(tag_s[i,j] \Leftrightarrow tag_t[i,j])}{\sum_{i,j} Count(tag_s[i,j])}$$



(a) parse tree in source Treebank



(b) conversion tree with target Treebank grammar

Figure 1: source tree and its conversion tree with target grammar formalism

Then Unlabeled aligned constituents F1 score (UAF) is defined as:

$$Score(p_s, p_{s\to t}) = \frac{2 \times UAP \times UAR}{UAP + UAR}$$
$$= \frac{2 \times \sum_{i,j} Count(tag_s[i,j] \Leftrightarrow tag_t[i,j])}{\sum_{i,j} (Count(tag_s[i,j]) + Count(tag_t[i,j]))} \quad (1)$$

(2) Labeled aligned constituents F1 score (LAF)

In the last subsection, we define $Score(p_s, p_{s\to t})$ according to UAF. In fact, the tags of constituents bring us much information to score conversion trees. So we define $Score(p_s, p_{s\to t})$ with Labeled aligned constituents F1 score (LAF) in this subsection.

Because the annotation schemes are different, constituent tags in source Treebank may be much more different from target Treebank. The number of such tags may be drastically different and the mapping may not be one-to-one. To eliminate the contradiction, we assume that each tag in source Treebank can be converted into every tag in target Treebank with various probabilities. So there is a converting matrix representing the converting probabilities, and we can calculate the converting matrix through source Treebank and *N*-best conversion trees.

Given the source Treebank and N-best conversion trees, first we align all the constituents, then collect all the aligned tags and compute the converting probability as the following equation.

$$p(tag_s \to tag_t) = \frac{Count(tag_s \Leftrightarrow tag_t)}{Count(tag_s)} \quad (2)$$

Finally, we modify UAF computed by equation (1) into LAF as below.

$$Score(p_s, p_{s\to t}) = \frac{2 \times \sum_{i,j} (1 + \gamma \times p(tag_s \to tag_t)) \times Count(tag_s[i,j] \Leftrightarrow tag_t[i,j])}{\sum_{i,j} (Count(tag_s[i,j]) + Count(tag_t[i,j]))} \quad (3)$$

In equation (3), γ is a tunable variable, which is used to weight the converting probability. Especially, LAF will be transferred into UAF when γ=0.

3.3 Corpus weighting technique

In line 12 of Algorithm 1, we train a new parser with target Treebank and conversion trees. However, the errors in automatically conversion trees are unavoidable and they would limit the accuracy of the self-trained model. So we have to take some measures to weight the gold target Treebank and the automatically conversion trees. McClosky et al. (2006) and Niu et al. (2009) take the strategy that duplicates the gold Treebank data many times. However, this strategy isn't suitable for PCFG-LA parser [1] (Matsuzaki et al., 2005; Petrov et al., 2006), because PCFG-LA employs an EM algorithm in training stage, so duplicating gold Treebank would increase the training time tremendously. Instead, according to Huang and Harper (2009), we weight the posterior probabilities computed for the gold and automatically converted trees to balance their importance.

Let $count(A \to \beta \mid t)$ be the count of rule $A \to \beta$ in a parse tree t. T_t and $T_{s\to t}$ are the sets of target Treebank and automatically converted trees from source Treebank respectively. The posterior probability of rule $A \to \beta$ (with weighting parameter α) can be expressed as:

[1] We will use BerkeleyParser as our baseline parser, which is a PCFG-LA based parser.

Feature templates
The label of the current constituent;
The label of the left most child, the middle child and the right most child;
The head word of the left most child, the middle child and the right most child;
The POS tag of the head word of the left most child, the middle child and the right most child;
Bigram of label, head word and POS tag of head word of the children: L/M, M/R;
Trigram of label, head word and POS tag of head word of the children: L/M/R;
The number of children;

Table 1: Feature Templates for Head Finding

$$p(A \to \beta) = \frac{\sum_{t \in T_t} Count(A \to \beta \mid t) + \alpha \sum_{t \in T_{s \to t}} Count(A \to \beta \mid t)}{\sum_{\beta} (\sum_{t \in T_t} Count(A \to \beta \mid t) + \alpha \sum_{t \in T_{s \to t}} Count(A \to \beta \mid t))} \quad (4)$$

4 Head Finding

In Task21 of CLP2010, we are required to find heads for each constituent. Our method is to make head finding as a post procedure after parsing.

We treat head finding problem as a classification problem, which is to classify each context-free production into categories labelled with their heads. For example, there are three types of heads: -0, -0-2 and -2 for $\text{vp} \to \text{vp wP vp}$, so we try to classify this production into categories labelled with -0, -0-2 and -2. First, we scan the train set and collect all the heads for each context-free production. Then we train a Maxent classifier to classify each context-free production into categories. We take the same feature templates for the classification as Chen et al. (2009) did, which is described in Table 1.

The head finding procedure proceeds in a bottom-up fashion, so that we can make use of heads of productions in lower layers as features for classification of the higher layers.

To evaluate the accuracy of our head finding method, we randomly select a development set, remove all head information and use our Maxent classifier to retrieve the heads. Experimental results show the accuracy has reached 98.28%. However, the final performance would drop much when the parse trees are generated automatically. Because the automatically generated parse trees would bring many errors, and the post procedure of head finding can't correct the errors.

5 Experiments and Analysis

5.1 Data Preparation

In order to evaluate the effectiveness of our approach, we do experiments for Chinese parsing using Tsinghua Chinese Treebank (TCTB) on target side and Penn Chinese Treebank (PCTB) on source side. We divide the training portion of the Tsinghua Chinese Treebank provided by CLP2010 into three parts as follows: 500 trees are randomly extracted as development set, another 500 as validating set and the rest trees are taken as training set. For trees in PCTB, all the empty-node and function tag information are removed. All the ParseVal measures reported in this paper are evaluated by the EVALB tool[2].

5.2 Experiments

In order to get a good final accuracy, we choose BerkeleyParser [3], which is a state-of-the-art unlexicalized parser, and train a model with the training set as our baseline. The F1 score of validating set parsed by baseline parser is 85.72%. In the following of this subsection, we try to combine our strategies into the baseline parser and evaluate the effectiveness. Because mult-time iterations can't improve parsing performance tremendously but cost much time during our experiments, we take *Iter*=1 here.

(1) Corpus weighting experiment

To evaluate the corpus weighting strategy, we take sentences (ignore the tree structure) in PCTB as unlabeled data, and train a parser with self-training strategy. F1 scores of validating set varying with α in equation (4) are shown in Figure 2. From Figure 2, we find that the F1 score varies with α, and reaches 86.46%

[2] http://nlp.cs.nyu.edu/evalb/

[3] http://code.google.com/p/berkeleyparser/

when α =1. The 0.74 absolute improvement comparing with the baseline certifies the effectiveness of our corpus weighting strategy.

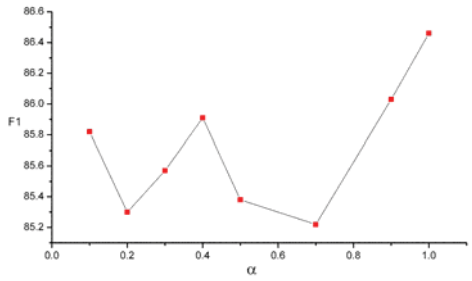


Figure 2: F1 score of self-training

(2) Parse selection experiments

In this subsection we evaluate our parse selection strategies with the help of PCTB. According to Algorithm 1, we train an initial parser with training set and development set. Then we generate 50-best parses list with the initial parser for each sentence in PCTB, and select a higher-score parse for each sentence through our parse selection strategies to build a conversion Treebank. Finally, we retrain a parser with training set and the conversion Treebank with the help of corpus weighting strategy.

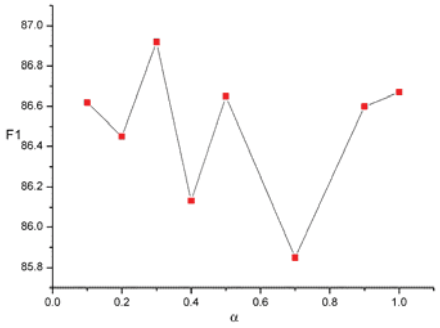


Figure 3: F1 score of UAF strategy

Figure 3 shows F1 scores of validating set using UAF to select higher quality parses. When α =0.3, F1 score reaches 86.92%. The improvement over baseline is 1.2 percentage points. Comparing with the highest F1 score of self-training, we got 0.46 more improvement. So our parse selection strategy with UAF is effective.

Because the highest F1 score is at the point α =0.3 in Figure 3, we choose α =0.3 to evaluating LAF strategy. Figure 4 shows F1 scores on validating set using LAF. The highest F1 score is 87.44% at the point γ =6, and it gets 1.72 percentage points improvement over baseline. Comparing with UAF, LAF gets 0.52 more improvement. So we can conclude that the parse selection strategy with LAF is much more effective.

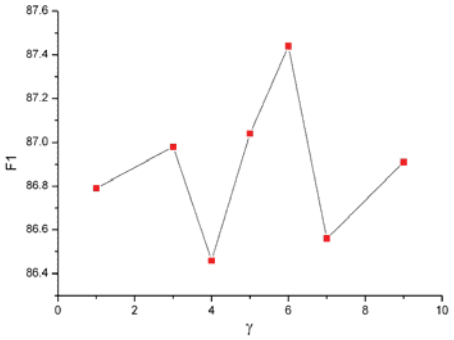


Figure 4: F1 score of LAF strategy

5.3 Discussion

Table 2 reports the highest performances of various strategies. From the table we can easily find that all strategies outperform the baseline parser. Corpus weighting experiment tells us that balancing the importance of gold target Treebank and conversion trees is helpful for the final performance. Using UAF to select conversion trees can get more improvement than self-training which just selects the best-first trees. This fact proves that our SSPTC strategy is reasonable and effective. Making use of LAF, we get more improvement than UAF. It tells us that exploiting source Treebank deeply can bring us more useful knowledge which is helpful to develop high-performance parser.

Strategy	F1 score
Baseline	85.72%
Corpus weighting	86.46%
UAF	86.92%
LAF	87.44%

Table 2: F1 scores of various strategies

6 Experiments for Task 2 of CLP2010

Task 2 of CLP2010 includes two sub-tasks: sub-sentence parsing and complete sentence parsing. For each sub-task, there are two tracks: closed track and open track. To accomplish tasks in closed track, we make use of our baseline parser shown in section 5 and train it with different parameters and data set. For open track, we make use of our SSPTC strategy and train it with different parameters and data set. We tuned the parameters on the development set and selected

Sub-task	Track	ID	Parser	Parameters	Train data
Sub-task 1	Closed	a	Berkeley	--	TS
		b	Berkeley	--	TS && VS
	Open	a	SSPTC	$\alpha = 0.3 \quad \gamma = 5$	TS && PTCB
		b	SSPTC	$\alpha = 0.3 \quad \gamma = 5$	TS && VS && PTCB
Sub-task 2	Closed	a	Berkeley	--	TS
		b	Berkeley	--	TS && VS
	Open	a	SSPTC	$\alpha = 0.3 \quad \gamma = 6$	TS && PTCB
		b	SSPTC	$\alpha = 0.3 \quad \gamma = 5$	TS && VS && PTCB
		c	SSPTC	$\alpha = 0.3 \quad \gamma = 5$	TS && PTCB
		d	SSPTC	$\alpha = 0.3 \quad \gamma = 3$	TS && PTCB

Table 3: The configurations of our systems. The abbreviations in the last column mean training set(TS) and validating set(VS) explaining in section 5.1.

some configurations which achieve higher performance on the development set(more details have been shown in section 5). The final parameters and training data of our systems are shown in Table 3[4]. We also make use of the approach explained in section 4 for the head finding procedure.

The parsing results of our systems on the test set can be found on the official ranking report. Our systems training with SSPTC strategy bring us an amazing performance which outperforms other systems in both the two sub-tasks.

7 Conclusion and Future work

In this paper, we propose a novel self-training strategy for parsing which is based on Treebank conversion. Benefiting from SSPTC strategy, we have gotten higher quality parse trees with the help of source Treebank, and gotten conversion Treebank with target Treebank grammar formalism simply and consistently. The parsing results on validating set show SSPTC is effective. We apply SSPTC to the test set of Task 2 in CLP2010, and get 1.27[5] percentage points improvement over baseline parser using the parameters tuned on validating set.

All the delightful results tell us that SSPTC is a promoting strategy for parsing. However, there is much knowledge in source Treebank remained to further exploit, e.g. the POS tags in source Treebank is a good resource to improve the POS tagging accuracy of target Treebank. So, in the next step we will exploit source Treebank deeply and try to get more knowledge from it for parsing.

Acknowledgement

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 6097 5053, 90820303 and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4, and also supported by the China-Singapore Institute of Digital Media (CSIDM) project under grant No. CSIDM-200804.

References

Eugene Charniak, 2000. A maximum-entropy-inspired parser. In NAACL-2000

Eugene Charniak and Mark Johnson, 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In ACL-05.

Xiao Chen, Changning Huang, Mu Li and Chunyu Kit, 2009. Better Parser Combination. In CIPS.

[4] The parsing result for system b in open track of sub-task1 has been submitted mistakenly, so the figures of this system on the official ranking report have no reference value.

[5] The F1 score of baseline parser is 75.24%, and it reaches 76.51% using TCBS strategy.

Michael Collins, 2003. Head-driven statistical models for natural language parsing. Computational Linguistics, 29 (4). pages 589-637.

M Collins, J Hajic, L Ramshaw and C Tillman, 1999. A statistical parser for Czech. In ACL-99. J Henderson, 2004. Discriminative training of a neural network statistical parser.

Zhongqiang Huang and Mary Harper, 2009. Self-Training PCFG grammars with latent annotations across languages. ACL-09.

R Hwa, M Osborne, A Sarkar and M Steedman, 2003. Corrected co-training for statistical parsers. Citeseer.

MP Marcus, B Santorini and MA Marcinkiewicz, 1993. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 19 (2). pages 313-330.

Takuya Matsuzaki, Yusuke Miyao and Jun'ichi Tsujii, 2005. Probabilistic CFG with latent annotations. In ACL-05.

David McClosky, Eugene Charniak and Mark Johnson, 2006. Effective self-training for parsing. In ACL-06.

Zheng-Yu Niu, Haifeng Wang and Hua Wu, 2009. Exploiting heterogeneous treebanks for parsing. In ACL-09, pages 46-54.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein, 2006. Learning accurate, compact, and interpretable tree annotation. In ACL-06.

Jong-Nae Wang, Jing-Shin Chang and Keh-Yih Su, 1994. An automatic treebank conversion algorithm for corpus sharing. In ACL-94.

Fei Xia and Martha Palmer, 2001. Converting dependency structures to phrase structures. In The 1st Human Language Technology Conference (HLT-2001).

Fei Xia, Owen Rambow, Rajesh Bhatt, Martha Palmer and Dipti Misra Sharma, 2008. Towards a multi-representational treebank. Proc. of the 7th Int'lWorkshop on Treebanks and Linguistic Theories (TLT-7). pages 207-238.

Qiang Zhou, 2004. Annotation Scheme for Chinese Treebank. Journal of Chinese Information Processing, 18 (004).